MSTROKE: METHODS OF FALL DETECTION AND DATA STORAGE

by

Austin Harris

Mina Sartipi
Professor of Computer Science
(Chair)

Dalei Wu
Professor of Computer Science
(Committee Member)

Yu Liang
Professor of Computer Science
(Committee Member)

MSTROKE: METHODS OF FALL DETECTION AND DATA STORAGE

by

Austin Harris

A Thesis Submitted to the Faculty of the University of Tennesse at Chattanooga in Partial Fulfillment of the Requirements of the Degree of Master of Science: Computer Science

The University of Tennessee at Chattanooga

Chattanooga, Tennessee

December 2017

ABSTRACT

Stokes are the leading cause of disability in adults in the United States. Falls are prevalent at all stages of recovery among post-stroke patients, and falls can cause serious or life threatening injuries. In this thesis, multiple fall detections methods are explored in order to minimize the faller's wait time. This research is an extension to our research on mStroke, a real-time and automatic mobile health system for post stroke recovery and rehabilitation. The proposed system consists of an application (mobile app) that is paired with bluetooth low energy (BLE) modular sensor devices. The sensors provide real-time acceleration, and gyroscopic data to the mobile application. This data is used to classify fall and non-fall activities performed by the user. The focus of mStroke has been on front-end development of application features. To address back-end long-term storage, a data storage solution for mStroke is investigated.

ACKNOWLEDGMENTS

First and foremost, I want to first thank all of family who have shown me support throughout my time as a graduate student.

I want to acknowledge my committee members, Dr. Yu Liang and Dr. Dalei Wu.

Finally, I want to express my appreciation to my advisor and mentor, Dr. Mina Sartipi. Her guidence over the past years has been invalueable.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 mStroke

Strokes are the leading cause of disability in adults. 795,000 people experience a stroke in the United States anually [**?**]. Strokes can lead to decreased independence, increased health care costs, and death. In the United States, someone dies from a stroke every four minutes [**?**].

In order to improve stroke rehabilitation and diagnosis the research group I work with developed a system called mStroke. mStroke aims to improve stroke-care outcomes and rehabilitation by analyzing data collected from stroke patients during and after rehabiliation. Previously this data was almost non-existant. Our application collects this data needed to correlate successful versus unsuccessful recoveries. Using this data rehabilitation techniques can be analyzed to imrove patient outcomes.

mStroke is a real-time automatic mobile health system for post-stroke recovery and rehabilitation. The system in combination with body-worn sensor network can be sent home with patients for use between therapy sessions. As patients use the system, data is collected based on their performance of each test. The results of these tests are stored and can provide the therapist with a better understanding of how the patient is recovering. Therefore, the therapist can further personalize the patient's rehabilitation to his/her needs.

In this thesis two main features for the mStroke system will be explored: fall detection and data storage. First, a system is investigated that can detect falls while a patient is using the mStroke system. Two main approaches were taken to accomplish this issue: rule-based and machine-learning based fall detection. In addition to developing a means of fall detection I also explore the effect sensor location and the number of sensors has on the detection performance. Lastly, mStroke's current implementation does not have a solution for long term data storage. Due the Health Insurance Portability and Accountability Act (HIPAA), specialized data storage is needed. Prior to my reserach, the therapist may take notes on

paper of the patient's scores and performance after. To address this issue, I will explore how to design a 'backend' HIPAA compliant data storage solution for mStroke.

Figure 1.1 mStroke's graphical user interface

1.1.1 Hardware

The mStroke system consists of an iOS application used in conjunction with wireless sensors. The sensor used was designed by Variable Inc and is called the Node Inertial Measurement Unit (IMU). The Node was chosen due to its low-energy requirements and low-latency. The base platform of Node consists of a tri-axil accelerometer, magnetometer, and gyroscope. Figure 1.2 shows the Node sensor, and its accleration vectors. The Node commicates with our iOS application using the Bluetooth 4.0 Low Energy (BLE) protocol.

The Node can transmit motion data to the iOS application at up to 120 per second. The number of sesnors and their location used depends on which test is being performed.

1.1.2 iOS Application

The mStroke system was developed in the Swift programming language and is able to run on any iOS device including iPads and iPhones. Previously Brandon Allen, Brian Williams, and Robert Derveloy have worked on developing different components of the mStroke application. When I began my research, the application was implmented in the Objective-C programming language. I decided to port the application to Swift from Objective-C. Swift has many advantages to Objective-C including: performance improvements, open-source, and is less error-prone due to its compact syntax in comparison to Object-C.

Recently the apps graphical user interface was redesigned by Rebekah Thompson and is now specifically designed for the iPad. This new interface was designed in order to make the application more user-friendly for elderly users. Colors, font, font-size and the flow of the applicaiton were all included in the improvements. The newest version of the application's graphical user interface can be seen in Figure 1.1.

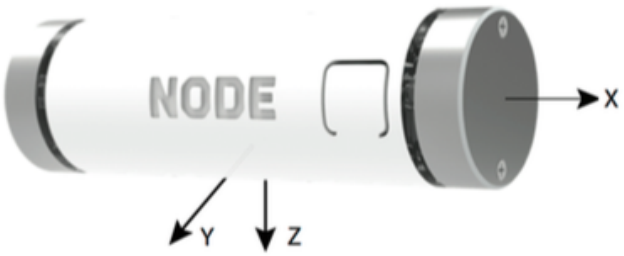

Figure 1.2 Variable Inc's Node with Aceeleration Axis

1.2 Existing Features

1.2.1 Functional Reach Test

The functional reach test (FRT) is used to evaluate a patient's stability. In order to perform the functional reach test using mStroke two sensors are needed. One sensor is attached to the wrist, and one sensor is attached to the chest as shown in figure 1.3. To begin the test the user should stand up and raise their arm with the sensor attached parrallel (90 degress) to the ground. Before the test can begin, the app ensures that the user is in the proper starting position using the effect of gravity on the acceleration values from each Node. If the user is in the correct position the app allows the user to begin the test. Once the start button has been selected that user must reach forward, bending at the waist. The user should not reach beyond their limits. The goal is to reach the maximum distance they can while remaining in the standing postion. Once the user has reached the farthest distance they can reach the stop button should be pressed. The app will then display the furthest distance reached during the exam. The distance is then compared to the norms of the user's age group which rates the user's margin of stability.

Figure 1.3 Subject wearing the FRT system

1.2.2 Motor Arm and Motor Leg

The Motor Arm and Motor Leg tests are components of the National Institute of Health Stroke Scale (NIHSS). Physicians use the NIHSS Stroke Scale as a tool in order to give a quantitative measure to the effect a stroke has on a patient. For both tests a single sensor is needed. The sensor is placed on the upper arm for the Motor Arm test, and attached to the lower shin for the Motor Leg test.

In order to perform the Motor Arm test the patient should be sitting down. The area below the arm should be clear of objects so the patients arm can hang straight down and can be raised to ninety degrees without interference. To begin the test the subject's arm should be perpendicular to the ground, hanging straight down. The physican will press the the start button on the app, which plays a sound to let the physican know the app is ready to begin the test and the user's arm is in an acceptable position. Next, the subjecst arm is raised by the physician until the arm is parrellel to the ground or at 90 degrees. When the physician has reached 90 degrees a sound is played to let the physician know to let go of the subject's arm. The goal is for the subject to hold their arm in this position for the extent of the test. After 10 seconds, if the arm is in the exact location the subject receives a score of 0. If the user is able to resist gravity for the entirety of the test, but the arm drifts down slowly the score is a 1. If the subject is able to resist gravity for a portion of the test, but the arm reaches the resting position before the end the score is 2. Lastly, if the subjects arm displays no resistance to gravity and immediately falls to its original resting state the subject is asked to perform a second test. In the second test the subject is asked to raise their arm from the resting position with no assistance. The subject receives a score of 3 if there is any arm movement, and a 4 if no movement is observed.

The Motor Leg test is very similar to the Motor Arm test. The sesnsor is attached to the subjects dominant leg, and the subject is laying down instead of sitting. The test lasts for 5 seconds, and the leg is raised to 30 degrees from its initial position parallel to the ground. The scoring system remains the same.

1.2.3 Gait

The Gait test determines how well a patient walks. In a clincial setting, the physician starts a timer and the patient is asked to walk a predetermined distance at a comfortable

speed. When the patient reaches the end of the disntace the timer is stopped. Nomative data from patients of the same class of the patient is used to determine the patients performance.

The Gait algorithm uses angular calculations similar to the Motor Arm and Motor Leg tests. The change in the angles of the patient's legs while walking is recorded, and in combination with the length of the patients legs can be used to estimate the distanc walked.

CHAPTER 2

MACHINE LEARNING BASED FALL DETECTION

2.1 Introduction

Falls are common for the elderly and individuals with decreased independence or functional limitations [1]. For example, one third of people above 65 years of age have a fall every year [1]. Falls are also prevalent in stroke survivors at all stages of recovery [2], [3], [4], [5], [6]. Community dwelling individuals with chronic stroke ave the highest fall incidence at 46% [5]. These falls can cause serious or even life threatening injuries, such as hip fracture and head tauma [1]. Falls can also result in progressive activity and participation limitations, increased fear of falling, and depression. All these consequences compound to negitively impace fallsers' qualities of life and increase caregivers' burdens.

In order to prevent falls as much as possible and mitigate the corresponding consequences when falls do occur, fall risk estimation and fall detection can be exploited. Fall risk estimation attempts to predict fall probability in the near future while fall detection tries to identify a fall situation as quick as possible [1]. We have studied real-time fall risk estimation for individuals post-stroke using the functional reach test, explained in Section 1.2. [7], [8], [9]. In this section, I will explore fall recognition using wearable technologies and machine learning algorithms, which have a great potential for clinical use in post-fall follow-up and prescribed intervention.

2.2 Related Work: Machine Learning

Wearable technologies have been employed for activity recognition and fall detection [10], [11], [12], [13], [14], [15]. These papers considered wearable motion sensors or inertial measurement sensors, such as accelerometer, gyroscope, magnetometer, or their combinations. Beyond fall detection, the determination of fall direction can provide more information about

fall context and severity [16]. Several manuscripts have been published, explicitly using wearable sensors for fall direction identification [15], [17]. In order to further improve the accuracy and reliability of fall detection and fall recognition, more than one wearable sensor can be applied, which leads to a new design problem of sensor placement/location [18], [19], [13]. In terms of data analytics, rule-based and machine learning-based methods are two main categories. In this section, we are more interested in machine learning-based methods due to their powerful modeling ability of complex relationships between input data and output classed and the corresponding discriminative capability for multi-class classification.

2.3 Summary of Technical Contribution

In this chapter, we investigate fall recognition including fall detection and fall direction identification by taking advantage of wearable technologies and machine learning algorithms. Secondly, motion sensor placement is another main focus of the study. Three potential locations for wearable sensors were considered. Fall recognition performances were demonstrated based on different feature selection approaches, different supervised learning algorithms, different sensor configurations and different feature numbers. Our exhaustive research justifies the accuracy and reliability of the proposed fall recognition system and reveals the effects of sensor placement and the feature number on the recognition performance.

2.4 System Design

Up to three wearable sensors and an iOS application were used to acquire raw motion data related to different activites under investigation. As mentioned in Section 1.1.1, our sensor collects accelerometer, gyroscopic, and magnetomic data. This data was stored on the ipad running a custom built iOS application made specificly for data collection. The sensor placement is a key design dimension in our study, taking into account recognition performance and usage comfort. Three potential locations, i.e., sternum, waist, and right leg, are considered for wearable sesnor locations. Raw motion data was collected from all three sensors simultanesouly for the following data analytics to detect and recognize falls.

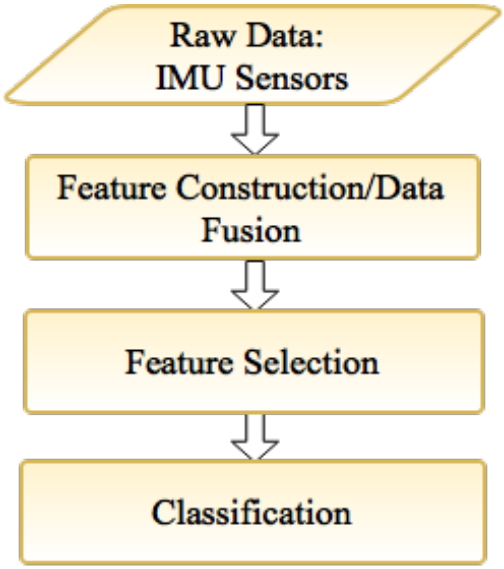

Figure 2.1 Fall classification process

2.5 Data Analytics

There are three main steps in data analytics, i.e., feature construction and data fusion, feature selection, and classification [20].

2.5.1 Feature Construction and Data Fusion

In our study, 54 statistical features are calculated from the collected raw data for each wearable sensor to build the feature vector based on tri-axial accelerometer data and tri-axial gyroscope data shown in Figure 2.2. These statistics include range, mean, absolute mean, mean cross-rate, zero cross rate, standard deviation, covariance, and mean trend. If more than one sensor is considered for our model, the corresponding feature vectors will be concatenated together, which means the size of featurevector for two or three sensors will be 108 or 162, respectively.

2.5.2 Feature Selection

Feature selection is a commonly used approach in dimensionality reduction. Feature selection searches for a subset of the original features directly from the high-dimensional feature

chestAccelX	chestAccelY	chestAccelZ
-0.009766	0.518571	-0.822291
-0.009766	0.518571	-0.822291
-0.027833	0.533708	-0.873074
0.052248	0.34962	-0.863308
0.052248	0.34962	-0.863308
0.052248	0.34962	-0.863308
-0.046876	0.308115	-0.973663
-0.046876	0.308115	-0.973663
-0.046876	0.308115	-0.973663

Figure 2.2 Format of the raw data received from the Node.

vector. Generally speaking, feature selection can reduce the space required to store feature data, simplify classification/recognition model, and enhance generalization by reducing model overfitting [21]. When we implement machine learning algorithms in mobile or resource constrained devices, the advantages of feature selection are more significant. Memory usage is reduced with decreased feature size. Energy usage is also reduced and runtime is shortend due to the low computational complexity of the simplified model, which makes machine learning suitable for for more long duration real-time applications. In theory, feature selection approaches can be divided into three categories, i.e., filter based approaches, wrapper-based approaches, and embedded-based approaches. [21] Here, we focus on three filter-based approaches, i.e., SelectKBest, ReliefF, and Robust feature selection. [22], [23], [**?**] Based on a feature selection approach, we can obtin a feature importance score (or weight) for each feature. SelectKBest generates feature rank based on univariate statistical tests. [21] ReliefF uses a kNearestNeighbor algorithm to determine feature rank, while our robust feature selection uses a combination of techniques to aggregate multiple feature rank scores into a single score. The features with high scores are selected to build the new feature vectors for the final step of classification.

2.5.3 Classification

Different supervised learning algorithms are applied to build classification models for fall recognition. Such algorithms include Random Forest, Linear Support Vector Machine

(LSVM), Logistic Regression, and k-Nearest Neighbors.

Random Forest classification falls into a category of learning algorithms known as "ensemble learning". These algorithms produce many classifiers and aggregate the results. [24] Each classification tree is fit to a randomly selected set of input data. To predict new observations the algorithm combines the predictions from all the trees. [25]

The Linear Support Vector Machine classifier seperates training data with a hyper-plane. [26] Its goal is to determine the optimal seperating hyperplane among classes. The optimal seperating hyperplane is the linear classifier that seperates each class, and has the greatest distance from itself to the nearest data points of each class. [27]

Multi-Class Logistic Regression creates "new" training sets where each class is divided into subsets along with all other classes in order to form a binary classification problem. A binary classifier is fitted for each subset of training data. These binary models are used to compute the probability of a new observation's class. [28]

The k-Nearest Neighbors classifier determines the class of a new observation feature vector by evaluating the class of "nearest" neighbors. Out of the "nearest" neighbors, the class with majority is chosen for the class of the new observation. [29]

2.6 Proof of Concept

As a proof of concept, an initial in-lab experiment was designed before subjects were recruited for the official testing. Data was collected using three subjects. The experiment considered nine activities: sitting/standing, laying down, walking, walking up-stairs, walking down-stairs, fall left, fall right, fall forward, and fall backward. Each subject performed each activity fifty times. The total dataset consisted of 1,350 entries. As explained above, raw acceleration and gyroscopic data was collected during these activites. A fifty-four feature vector was computed for each activity including features such as: range, mean, absolute mean, mean cross rate, zero cross rate, standard deviation, convariance, and mean trend.

For testing purposes, only the Random Forrest and Support Vector Machine classification algorithms were used. The results of the Random Forrest classifier using the ReliefF feature selection technique and the Support Vector Machine (SVM) using the SelectKBest feature selection technique are shown in Figure 2.3 and Figure 2.4, respectively. The combination of the Random Forrest classifier was able to achieve an accuracy of 98.0%, while the SVM

was able to achieve an accuracy of 98.1%. Figure 2.5 and Figure 2.6 show the accuracy of the classifiers based on the number of features.

Feature Selection: ReliefF			Number of Features: 41						Accuracy: 98%	
	Sitting /Standing	Lying Down	Going Downstairs	Going Upstairs	Walking	Fall Left	Fall Right	Fall Forward	Fall Backward	All
Sitting/Standing	149	0	0	0	0	0	1	0	0	150
Lying Down	0	150	0	0	0	0	0	0	0	150
Going Downstairs	0	0	146	4	0	0	0	0	0	150
Going Upstairs	0	0	16	134	0	0	0	0	0	150
Walking	0	0	0	0	150	0	0	0	0	150
Fall Left	0	0	0	0	0	149	0	1	0	150
Fall Right	0	0	0	0	0	0	149	0	1	150
Fall Forward	0	0	0	0	0	1	0	149	0	150
Fall Backward	0	0	1	0	0	0	2	0	147	150
All	149	150	163	138	150	150	152	150	148	

Figure 2.3 Results of Random Forrest classifier for each combination of features

Feature Selection: SelectKBest			Number of Features: 35						Accuracy: 98.1%	
	Sitting /Standing	Lying Down	Going Downstairs	Going Upstairs	Walking	Fall Left	Fall Right	Fall Forward	Fall Backward	All
Sitting/Standing	150	0	0	0	0	0	0	0	0	150
Lying Down	0	150	0	0	0	0	0	0	0	150
Going Downstairs	0	0	149	1	0	0	0	0	0	150
Going Upstairs	0	0	2	142	6	0	0	0	0	150
Walking	0	0	0	4	146	0	0	0	0	150
Fall Left	0	0	1	0	0	148	1	0	0	150
Fall Right	0	0	0	0	0	0	149	0	1	150
Fall Forward	0	0	0	0	0	0	1	149	0	150
Fall Backward	0	0	1	0	0	0	2	6	142	150
All	150	150	153	147	150	150	152	150	148	

Figure 2.4 Results of Random Forrest classifier for each combination of features

The results of the in-lab experiment confirm our goal of classifying fall versus non-fall activites. The structure of this experiment will be in the next section as a guide to develop a classifier from a larger dataset.

2.7 Experimental Procedure and Protocol

Fourteen healthy subjects participated in the experiment with an Institutional Review Board approval from the University of Tennessee at Chattanooga. These subjects included

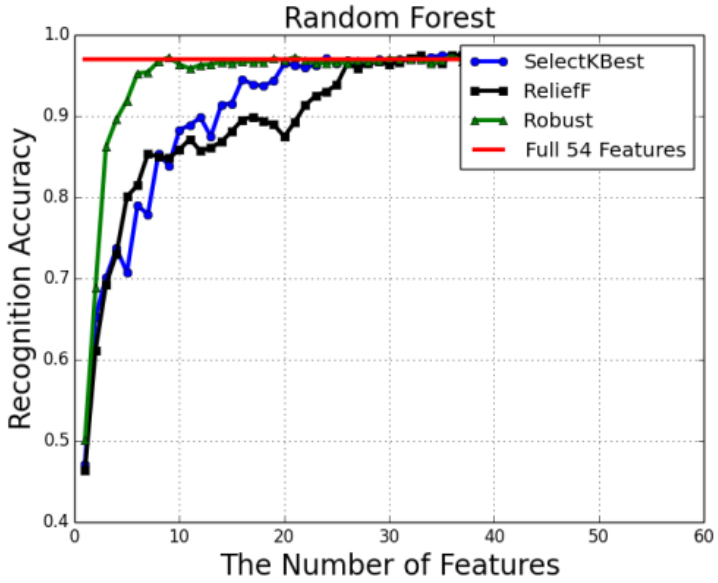

Figure 2.5 Results of Random Forrest classifier for each combination of features

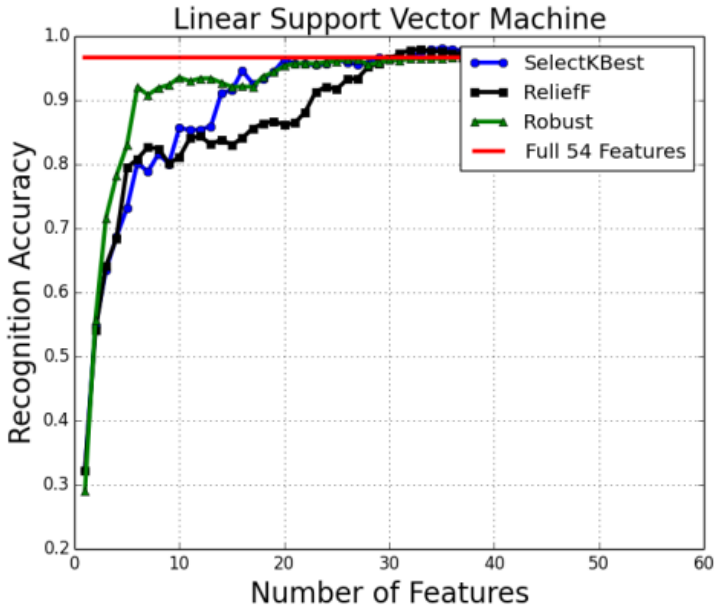

Figure 2.6 Results of Support Vector Machine for each combination of features

7 males and 7 females. Their ages ranged from 22 to 50, with an average age of 27. Their heights ranged from 154.9cm to 190.5cm, with an average height of 165.1cm. Subjects were outfitted with three sensors on their chest, waist, and leg as shown in figure 2.4. Each

subject was asked to perform a total of 21 activities. Each activity was randomly chosen from three activities of daily living (i.e. reaching up, reaching down, and walking), and four fall directions (i.e. fall forward, fall backward, fall right, and fall left). In terms of fall recognition, we grouped three activites of daily living as one class. Thus, we had a total of 5 identification classes (i.e. non-fall, fall-left, fall-right, fall-forward, and fall-backward).

Figure 2.7 Three sensor locations considered

2.8 Results

Leave-one-subject-out cross validation was utilized to split the whole dataset into the training and testing sets. Python Scikit-learn framework was used to build different classification models based on different supervised learning algorithms, different feature selection approaches, different sensor configurations, and different feature numbers. The best recognition performances in terms of classification accuracy together with the corresponding feature numbers (i.e. the input dimensions of classification models) are shown in Table 2.1, Table

Table 2.1 Recognition Accuracy (The Number of Features) using Random Forest

Sensor Configuration	SelectKBest	ReliefF	Robust
Sternum	92.5% (38)	93.9% (48)	93.9% (35)
Leg	93.5% (28)	94.2% (32)	93.9% (41)
Waist	95.2% (50)	95.2% (20)	95.6% (33)
Sternum+Leg	96.9% (86)	97.3% (68)	97.6% (12)
Sternum+Waist	97.3% (86)	96.9% (68)	97.3% (12)
Leg+Waist	96.3% (86)	96.6% (68)	96.6% (12)
Sternum+Leg+Waist	98.3% (72)	98.3% (127)	98.3% (94)

Table 2.2 Recognition Accuracy (The Number of Features) using Linear Support Vector Machine

Sensor Configuration	SelectKBest	ReliefF	Robust
Sternum	94.2(52)	94.2% (50)	94.6% (27)
Leg	94.9% (24)	94.6% (43)	94.6% (44)
Waist	95.9% (39)	95.9% (46)	96.9% (34)
Sternum+Leg	96.9% (92)	97.6% (105)	97.3% (60)
Sternum+Waist	96.9% (92)	97.2% (105)	97.6% (60)
Leg+Waist	97.3% (92)	96.6% (105)	97.9% (60)
Sternum+Leg+Waist	97.9% (24)	97.3% (150)	97.9% (24)

2.2, Table 2.3, and Table 2.4 for Random Forrest (100 trees), Linear Support Vector Machine, Logistic Regression, and k-Nearest Neighbors (k=5), respectively. All the recognition accuracies are above 90% and can reach as high as 98.3%. Generally speaking, the performances using more than one sensor are better than those using one single sensor. Three sensors used together are more likely to give the best performance. If only one sensor is considered, this sensor should be put on the waist. Meanwhile, all recognition accuracies can achieve their best using less selected features instead of full constructed features, which verifies the importance of feature selection.

2.9 Conclusion

Based on our newly developed mStroke, we propose a fall recognition system using wearable technologies and machine learning algorithms. Three wearable sensors were utilized for data acquisition. Three feature selection approaches and four supervised learning al-

Table 2.3 Recognition Accuracy (The Number of Features) using Logistic Regression

Sensor Configuration	SelectKBest	ReliefF	Robust
Sternum	93.9% (46)	93.9% (53)	94.6% (33)
Leg	93.5% (37)	93.5% (47)	92.5% (28)
Waist	95.6% (49)	95.1% (29)	95.2% (26)
Sternum+Leg	96.3% (49)	96.7% (24)	96.9% (27)
Sternum+Waist	96.6% (58)	96.6% (28)	96.5% (27)
Leg+Waist	97.6% (18)	96.3% (34)	96.9% (28)
Sternum+Leg+Waist	98.3% (25)	97.6% (33)	97.9% (39)

Table 2.4 Recognition Accuracy (The Number of Features) using k-Nearest Neighbors

Sensor Configuration	SelectKBest	ReliefF	Robust
Sternum	90.8% (44)	91.8% (19)	93.9% (38)
Leg	93.5% (25)	92.9% (25)	93.5% (14)
Waist	93.9% (50)	94.9% (16)	94.9% (36)
Sternum+Leg	96.9% (74)	96.6% (47)	97.6% (29)
Sternum+Waist	95.6% (74)	96.3% (47)	95.9% (29)
Leg+Waist	95.9% (74)	95.9% (47)	95.9% (29)
Sternum+Leg+Waist	97.3% (28)	97.6% (85)	97.3% (68)

gorithms were explored for data analytics to detect and recognize falls. The experimental results showed accurate and reliable recognition performances. Meanwhile, sensor placement and the feature number were highlighted as important design dimensions.

CHAPTER 3
THRESHOLD-BASED FALL DETECTION

3.1 Introduction

In additon to the study in Chapter 2, a threshold-based fall detection algorithm will be investigated. The need for real-time fall detection systems was covered in Section 2.1. The machine learning approach discussed in Chapter 2 provided promising results, however, the classifier does not yet contain a system for real-time raw date segmentation. Implementing the classification system is possible, but a simple sliding window scheme would need to be used to segment real time data. Using a simple sliding-window technique to segment the data creates a high demand for processing power and energy consumption. Due to these high processing power and energy consumption requirements, we investigate the reliability of a threshold based fall detection system.

In order to address the issue of response time and to prevent "long lie" ambient devices, surveillance systems and wearable technologies using inertial-measurement units have been deployed to detect falls. While visual-based detection systems offer high-accuracy results, our goal was to implment a mobile fall detection system that uses the current hardware of our system mStroke introduced in Section 1.1 [30]. Our focus is to extend mStrokes functionality to include a fall detection using the current sensor platform.

3.2 Related Work

Wearable technologies, vision based approaches, and ambient device based approaches have all been the subject of studies focused on fall detection systems. Our focus will be on the wearable technologies which have been studied extensively [10], [11], [12], [15]. Wearable devices, such as accelerometers and gyroscopes, were employed in order to detect falls. Acceleration data were collected during fall and non-fall events in order to determine thresh-

olds that signifiy a fall event. Threshold based fall detection systems are attractive due to their minimum processing requirements and low-energy consumption. We are interested in developing and implementing a rule-based fall detection algorithm in order to increase the safety of the community-dwelling individuals and in-home use of the mStroke system.

3.3 Summary of Technical Contribution

The goal of this study was to investigate whether body worn sensors used alongside a threshold-based fall detection application on a tablet can differentiate between fall and non-fall activities. The system focused on the use of acceleration data for the detection of falls. The system will consist a single sensor and an iOS application. The fall detection system considered two key components. First, the functionality to detect high-intensity activites. Secondly, differentiating falls activites from the these high-intensity activites.

3.4 Proposed System

The proposed fall detection system is an extension of our existing mobile health application mStroke introduced in Section 1.1. As discussed in Section 1.1, mStroke can run on any iOS device running iOS 10.1 or later. The fall detection system will use acceleration data only from mStroke's existing equipment to detect falls. No additional hardware will be needed in order to keep the portability and ease of use of mStroke.

The fall detection algorithm is executed in parrallel to mStroke's existing tests. Using mStroke, the user selects which test they would like to perform. Once a test is selected, the application displays the corresponding test's view. From this view the user can begin the test by selecting the start button. When the test begins, acceleration data are transmitted from the body worn sensors to the iOS device. If a fall is detected during any of the exams mStroke will stop the test and provide a on-screen notification asking if the user is okay.

The system uses the Nodes from Variable Inc as discussed in Section 1.2.1. The Nodes are connected via Bluetooth Low Energy. Each sensor contains a tri-axial accelerometer, gyroscope, and magnetometer. The data are transfered to the application at 30 Hz. The Node is shown in figure 2.1 along with its axis.

3.5 Fall-Detection Algorithms

For this study, there were two development iterations of the fall detection algorithm. Each iteration focused on develping the two main components of our fall detection algorithm: event intensity and posture analysis. The first iteration uses thresholds to detect high intensity movements such as falls, but cannot differentiate between falls and other high intensity movements. Therefore, fall detection using event intensity alone produces a system with a high false alarm rate. For example, if a patient squats from a standing position extrememly fast the acceleration may reach that of a fall even though they have not fallen. This issue is addressed by the second iteration which introduces a posture analysis component. Posture analysis aims to detect the posture of the patient prior to an high intensity event. If a high intensity event is detected and the patien is lying prone it is very likely they have fallen. The intensity analysis and posture analysis components are further investigated in the next sections.

3.5.1 Intensity Analysis Classification

The goal of this section is to discuss how to detect falls based on acceleration intensity. In order to establish a hypothesis, fall and non-fall activities were performed and raw acceleration data was collected. Because falls can occur in any direction, the direction of the acceleration is not considered. Thus, the magnitude was computed, using equation 3.1, using the acceleration vectors of all activities perfromed. Figures 3.1 and 3.2 show the linear acceleration for fall and non-fall activites. The linear acceleration of non-falls reached a maximum of $.80g$. Using this as a threshold, we can classify the majority of fall versus non-fall activities. The remaing error was produced was the result of high intensity non-fall activites.

$$|v| = \sqrt{x^2 + y^2 + z^2} \tag{3.1}$$

3.5.2 Posture Analysis

In order to improve our fall detection algorithm a posture analysis component was developed. This component determined the orientation of the subject post-fall. Knowing the orientation of the subject will allow us to counter the error produced from high-intensity

non-fall activites. The goal is to determine if the subject is lying down after a high-intensity activity has been detected. Raw acceleration data was used to determine the orientation of the user by computing the angle between each axis and the gravity vector.

Acceleration data are recorded in terms of gravity (g). Figures 3.1,3.2, and 3.3 show tri-axial acceleration data for a fall activity, and figures 3.5, 3.6, 3.7 show tri-axial acceleration data during an non-fall. The force of gravity created by the earth can be seen along the Z-axis for both the fall and non-fall data. During the fall activity show gravity can be seen transfering from the Z-axis to the Y-axis in Figures 3.1, and 3.3. Gravity measures approximately $1.0g$ along the vector perpendicular to the ground. This is known as the gravity vector. The gravity vector is always perpendicular to the ground. Therefore, using equation 3.2, we can compute the angle between the Z-axis and the gravity vector.

$$\phi = \arccos\left(\frac{A_y}{\sqrt{A_x^2 + A_y^2 + A_z^2}}\right) \tag{3.2}$$

The Node was attached with the power button on the left hand side and the button facing upward relative to the subject so that the Z-axis was perpendicular to the ground. When the subject is standing the Y and X axes read $0.0g$ and the Z-axis read $1.0g$. The angle between the gravity vector and the Z-axis is zero degress because the Z-axis is perpendicular to the ground when the subject is standing. By analyzing fall and non-fall data a the threshold for the angle to signify a fall was determined to be any angle greater than 45°. The angle must remain 45° for more than a single time interval. The time interval is determined by the sliding window discussed in the next section.

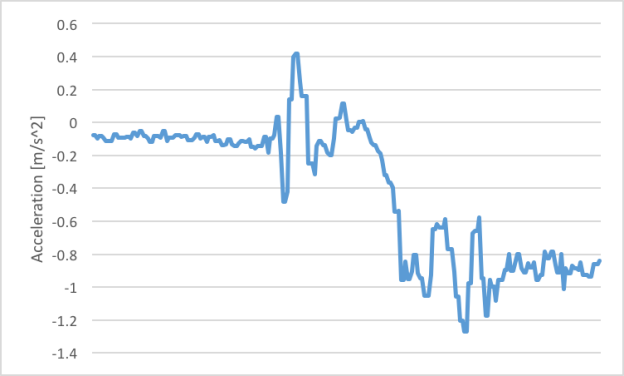

Figure 3.1 Acceleration along the Y-Axis during a fall event

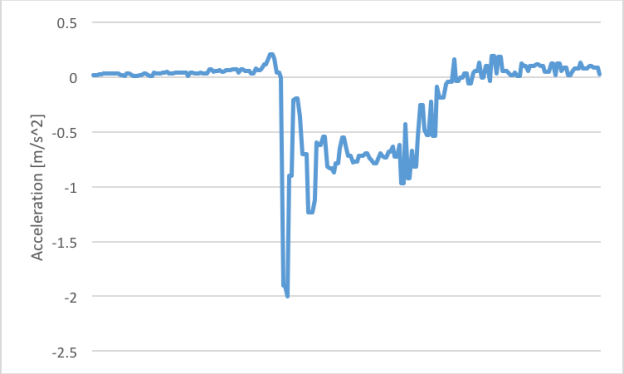

Figure 3.2 Acceleration along the X-Axis during a fall event

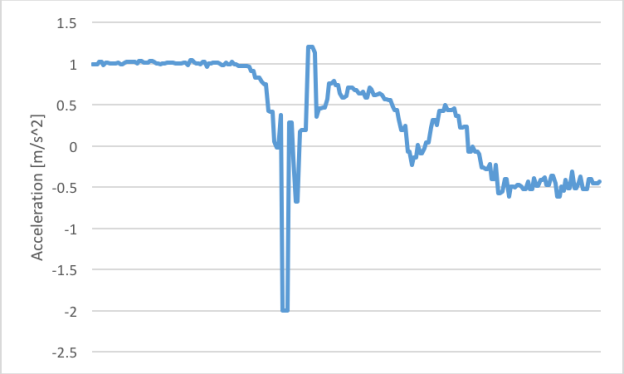

Figure 3.3 Acceleration along the Z-Axis during a fall event

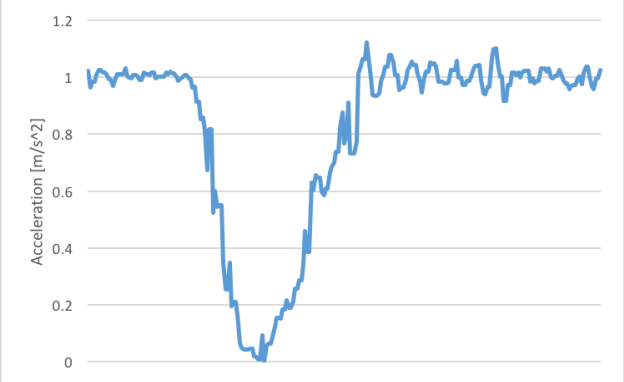

Figure 3.4 Acceleration along the Z-Axis during a non-fall event

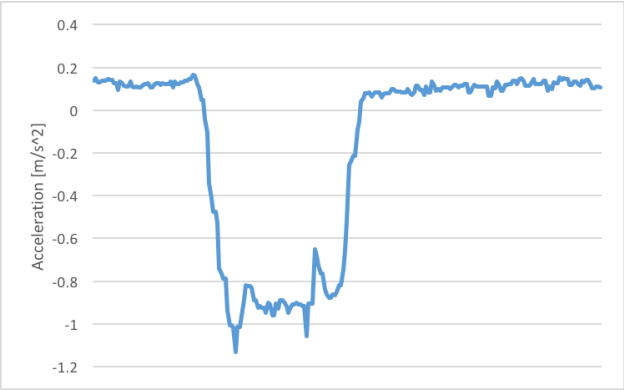

Figure 3.5 Acceleration along the X-Axis during a non-fall event

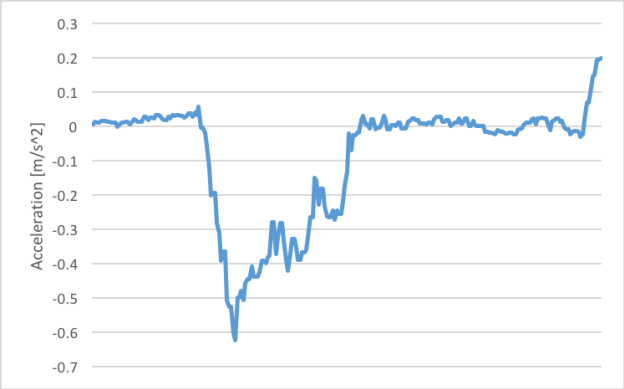

Figure 3.6 Acceleration along the Y-Axis during a non-fall event

3.5.3 Algorithm

The final fall detection algorithm consisted of both the intensity analysis and posture analysis components. Once the test had begun, the magnitude for each acceleration vector received was computed. Each magnitude was stored in a sliding window of fixed size. The size of sliding window stored eight seconds worth of readings from the sensor. If the change in magnitude exceeded the threshold of $0.8g$ posture analysis was performed. The Node is placed on the chest so that the Y-Axis is perpendicular to the ground and parallel to the subjects upper extrememity. As mentioned above, the angle that represents the subject's posture should be $0°$ if the subject is standing upright. If simultaneously, the change in magnitude exceeds $0.8g$ and the subject's chest exceeds a 45 degree angle a boolean value is set to true. If for the next five seconds the subject returns to an upright position, the boolean is set to false. Otherwise, a fall is detected.

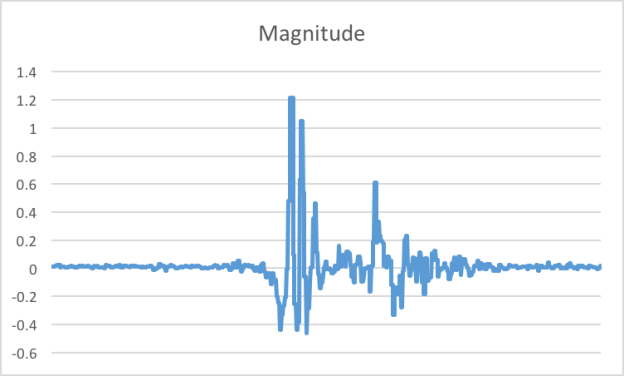

Figure 3.7 Magnitude of a fall event

3.6 Testing Methodology

In order to test the rule based fall detection we worked closely with our team from UTC's physical therapy department. Two seperate tests were performed to test each iteration of the fall detection algorithm. We obtained an IRB for each test, and each volunteer signed a release form.

For testing purposes, an iOS application was developed specifically for testing the fall detection algorithm. Before the exam, each subject was outfitted with a single sensor placed on their chest using velcro straps. Each subject was told prior to each test which activity they

would be performing. The physcial therapists would ask each subject if he/she understood what activity they were to perform. If the subject understood, the physical therapist would begin the test. Once the subject had performed their designated activity the user would press the stop button on the application. At this time, the application notifies the user whether a fall was detected or not. An additional observer recorded the apps results alongside what activity was observed.

3.6.1 First Iteration

The first iteration of the fall detection algorithm test consisted of 14 subjects (7 male, 7 female). Each subject was fitted with a Node that was attached to their sternum. Each subject performed 21 activites. Possible activites included: reaching up, reaching down, walking, and falling in four different directions. Each activity was evaluated individually. For each activity performed, a fall or non-fall result was reported by the application.

3.6.2 Second Iteration

The October tests consisted of 50 subjects (15 male, 35 female). Each subject was randomly assigned one of the protocols shown in figure 3.6. Each protocol consisted of three seperate activites followed by a fall or non-fall. Each activity was performed a total of three times. This test closely resembled our first test except non-fall and fall activites were performed within an individual test.

3.7 Results

3.7.1 Fall Detection

As discussed in the previous sections, the first iteration of the fall detection algorithm relies too heavily on pure acceleration thresholds. In order to counter the error produced, the posture analysis component was added to the fall detection system. The first iteration of the algoritm produced an accuracy of 85.0%, and 75.0% of the error was due to non-fall activites that consisted of high-intensity movements. After the addition of the posture analysis component, the second iteration of testing produced a much more desirable accuracy

of 92.7%.

3.7.2 Conclusion

This study enforced the hypothesis that a body-worn sensor can be used in combination with an mobile health application for fall detection. Our system provides real-time accurate and reliable fall detection solution. In the future, our system can be used to reduce "long lie" in fall scenarious. Therefore, reducing the negitive impact of falls on the elderly population.

CHAPTER 4

DATA STORAGE SYSTEM

4.1 Introduction

In this section we explore a data storage solution for the mStroke application. This study was conducted in close conjunction with Hector Suarez, a co-researcher at the University of Tennessee at Chattanooga. Hector brought his knowledge of information security in order to develop this data storage solution. In combination with my knowledge of data storage frameworks as well as iOS development we aimed to develop a secure and robust data storage solution.

Currently, no data are stored locally on the mStroke iOS application. As mentioned in previous sections, the system collects patient gyroscopic and acceleration data via wireless sensors that form a body sensor network. The system will first need to store general patient information such as name, address, age, weight, and height. Secondly, the system will need to store score data from NIHSS motor arm, NIHSS motor leg, gait, and the functional reach test. It is important that our design provide the functionality to allow physicians to analyze this data from a web portal.

Mobile devices are becoming ubiquitous for mobile health and rehabiliation. Securely integrating these devices into systems has become crucial to protecting sensitive data. Systems containing sensitive information are highly valuable to cyber criminals as the data can be illegally sold for profit. With increasing use of mobile devices, mobile health is making it possible for healthcare professionals to easily store, access, and analyze patient data. The Health Insurance Portability and Accountability Act introduced regulations that are meant to hold any entity that deals with protected health information (PHI) accountable for data breaches. Users may not always make the best security decisions so it is up to the developers to protect users and user data. This section presents an approach to designing and implementing a HIPAA compliant data storage solution for our mStroke system.

4.2 Summary of Technical Contributions

In order to develop a data storage solution for the mStroke system a base platform and infrastructure was developed. Using RESTful API's and existing database frameworks, a system was implmented that allowed user's data to be sent from the mStroke applcation to the database. The design of this platform needed to provide flexibility so that the security components could be added to the system. These security components provided local authentication via a biometric fingerprint authentication, created a permenant way to establish a secure encrypted communication with the server, provided remote authentication to the database and provide session management.

4.3 HIPAA Compliance

Health data that is individually identifiable is known as Protected Health Information (PHI). PHI is defined by the Health Insuance Portability and Accountability Act which was passed in 1996. One of the goals for HIPAA was to ensure the safe and secure storage of patient health data. HIPAA outlines administrative, physical, and technical regulations [31]. HIPAA regulations make sure that the confidentiality, integrity and the availability of medical records is enforced. The guidelines oulined by HIPAA were used to guide the developing the mStroke system.

4.4 System Design

In order to design a system that is flexible, a RESTful API was used to build the infrastructure needed to allow communication between the iOS application and the database. RESTful API's are used to design web-based applications by companies such as Facebook and Twitter. API stands for application programming interface, and REST stands for representational state transfer. The API will act as a middle man, and be used to allow our iOS application to speak to/store data into our database. The iOS applicaiton speaks directly to the RESTful API, and has no knowledge of underlying storage methods. The API handles all communication with the database. As data is pushed to the API from mStroke application, the API parses this data and stores it in the database. The data is sent to the web server in Javascript Object Notation (JSON) format, shown in Listing 4.1. Sails

provides a module that can be imported that configures communication with the database. The complete system can be visualized in Figure 4.1.

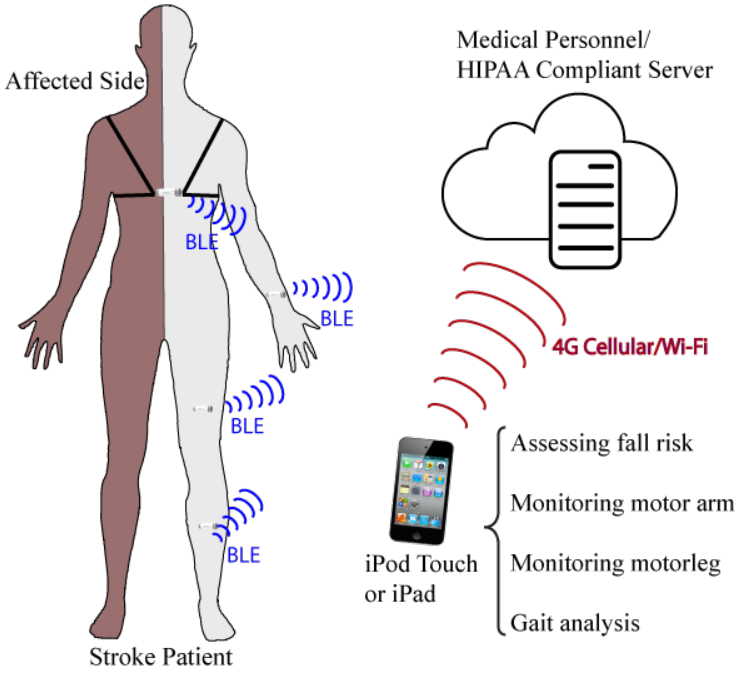

Figure 4.1 The complete mStroke system

mStroke's data storage is based on a client-server model. While designing this system, flexibility was an important attribute when choosing a web framework due to our system's security requirements. Our requirements consisted of an open-source web framework that was highly customizable. The Sails JS framework was chosen for the server side RESTful API. Sails is a Model View Controller (MVC) javascript framework that is built on top of Node JS. Its benefits include: the ability to use any database, API auto-generation, and is front-end agnostic. The document driven, NoSQL database MongoDB was chosen for actual data storage. The Sails REST API and the MongoDB database were installed on Ubuntu Server 16.04.

```
{
        "patientID": 182393,
        "testID": "FRT",
        "physicianID": 139,
```

```
5           "score": 12.3,
6           "date": "11/2/17"
7   }
```

Listing 4.1: Exampe of JSON data sent from the iOS application.

Using the Sails API, allowed us to implement our own features, and gives us the flexibility we need to implement our own security measures throughout the sytem. Sails provides a feature called Routes, Routes allow you to specify an "end-point" of a URL. When an "end-point" or Route is called, a corresponding function is executed. These Routes enabled us to provide direct calls that abstracted database CRUD operations (i.e. create, update, and delete) to store data and implement the security functionality required.

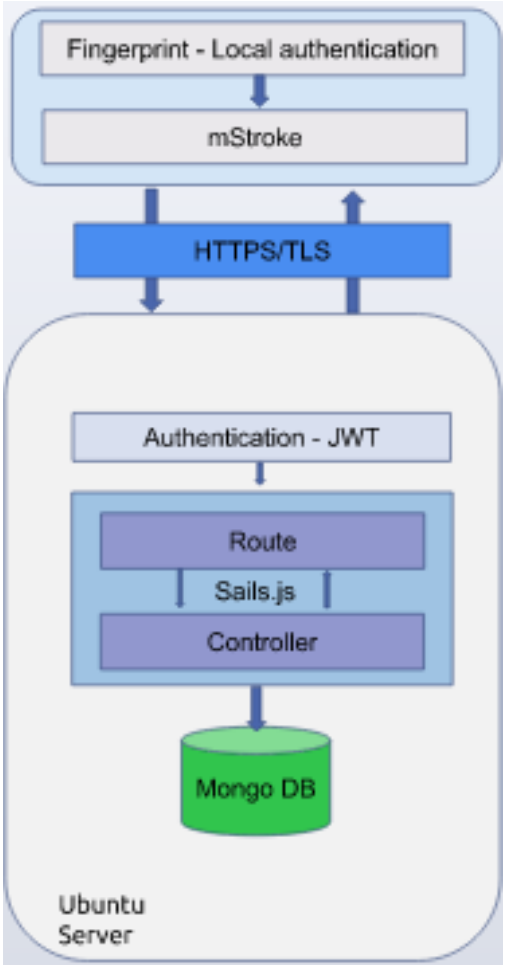

Figure 4.2 mStroke's data storage architecture

4.5 Security Components

In order to meet HIPAA regulations, four main security components were considered. A client-side biometric fingerprint authentication system was implmented in order to gain access to the applicaiton. This prevents un-authorized users from accessing the application, and uploading false information. Next, a transport layer security (TLS) certificate was generated, and the public key was manually pinned to the iOS application. This is used to encrypt client-server communication with TLS 1.2. The Sails framework provided a feature that allowed us to enforce a secure TLS connection. On the client-side, a function was implemented to verify the validity of the server using the pinned certificate. To manage sessions between the client and the server JSON web tokens were used. When the client-side application first requests access, the API verifies the application using login credentials. Next, a token is passed to the client, and once the client has the token it can push data to the server. Our system architecture was designed so that sessions only need to exist long enough to transmit the data. Therefore, each access token's expiration time was shortened to ensure the session could not be hijacked.

4.6 Conclusion

With the increasing use of mobile health applications, the need to secure mobile health systems has never been more important. The underlying system design provided a scalable, secure data storage solution for the mStroke application. Sails and MongoDB gave us the flexibility to implement the security components needed to meet HIPAA regulations. The back-end system of mStroke met the goal of allowing users to store their information in a secure manner. Once a test was performed, the application asks if the user would like to store his/her results. This data will allow physicians to: improve the quality of care, incease the ease of access, improve patient outcomes, and monitor patient progress in real-time.

4.7 Future Work

There has been extensive research on the benefits of electronic health records [32]. These benefits include: cost of storage in comparison to physical paper storage, an improved quality of care, and patient record mobility. With mStroke, patients and physicians will be able to

access healthcare records anywhere and anytime. A front-end web-based application with access control policy is the next step for the mStroke system. This web application will provide a variety of tools for data analysis. Lastly, it will allow physicians and patients to log in and asses the patients by viewing time series graphs and other metics representing the patients rahabilitation.

CHAPTER 5
CONCLUSION

5.1 Thoughts on Fall Detection

Both fall detection methods explored in this thesis met the expectations proposed. Both systems provided an accuracy above 90%. The next section will discuss future improvements of both systems. In addition to fall detection, the proposed data storage system is a great proof of concept for future studies. Our study proved existing components could be used as a foundation to a health data storage platform.

5.1.1 Future Work: ML-Based Fall Detection

A real-time raw data segmentation component is needed in order to complete the machine learning based fall detection system. Currently, activites were performed and stored as single data sets. This means the system was trained on many data sets divided by activity. This study developed a classifier that provided the accuracy needed to detect falls. Once a segmentation system is designed, the fall detection classifier can be implemented into the application.

5.1.2 Future Work: Threshold-Based Fall Detection

The fall detection system in its current implementation provides an accurate method of detecting falls. The addition of the posture analysis component provided the system with the means of differeniating falls from high-intensity activites. The mStroke system provides a viable fall detection system considering the limited high-intensity movements of the elderly.

5.1.3 Future Work: Data Storage

The data storage system was tested in the lab, however, certain production environment variables must be determined in order to scale the system. Certain production-level deployment scripts may be used in order to scale our system. Database design should be re-considered if the system is scaled beyond a single physcial therapists. The addition of databases, either No-SQL or SQL, can be added easily which allows the sytem to scale to multiple physicians. A single API can be used for multiple physicians. The API can direct data incoming from the application to it's corresponding database. The goals of this study were met, and implemented into our mStroke application.

REFERENCES

[1] D. Mozaffarian, E. J. Benjamin, A. S. Go, D. K. Arnett, M. J. Blaha, M. Cushman, S. de Ferranti, J.-P. Després, H. J. Fullerton, V. J. Howard *et al.*, "Executive summary: heart disease and stroke statisticsâ2015 update," *Circulation*, vol. 131, no. 4, pp. 434–441, 2015.

[2] A. Danielsen, H. Olofsen, and B. A. Bremdal, "Increasing fall risk awareness using wearables: a fall risk awareness protocol," *Journal of biomedical informatics*, vol. 63, pp. 184–194, 2016.

[3] A. Forster and J. Young, "Incidence and consequences offalls due to stroke: a systematic inquiry," *Bmj*, vol. 311, no. 6997, pp. 83–86, 1995.

[4] L. M. Wagner, V. L. Phillips, A. E. Hunsaker, and P. G. Forducey, "Falls among community-residing stroke survivors following inpatient rehabilitation: a descriptive analysis of longitudinal data," *BMC geriatrics*, vol. 9, no. 1, p. 46, 2009.

[5] A. A. Divani, G. Vazquez, A. M. Barrett, M. Asadollahi, and A. R. Luft, "Risk factors associated with injury attributable to falling among elderly population with history of stroke," *Stroke*, vol. 40, no. 10, pp. 3286–3292, 2009.

[6] L. Jørgensen, T. Engstad, and B. K. Jacobsen, "Higher incidence of falls in long-term stroke survivors than in population controls," *Stroke*, vol. 33, no. 2, pp. 542–547, 2002.

[7] A. Ashburn, D. Hyndman, R. Pickering, L. Yardley, and S. Harris, "Predicting people with stroke at risk of falls," *Age and ageing*, vol. 37, no. 3, pp. 270–276, 2008.

[8] B. Allen, R. Derveloy, K. Lowry, H. Handley, N. Fell, W. Gasior, G. Yu, and M. Sartipi, "Evaluation of fall risk for post-stroke patients using bluetooth low-energy wireless sensor," in *Global Communications Conference (GLOBECOM), 2013 IEEE*. IEEE, 2013, pp. 2598–2603.

[9] B. Williams, B. Allen, Z. Hu, H. True, J. Cho, A. Harris, N. Fell, and M. Sartipi, "Real-time fall risk assessment using functional reach test," *International journal of telemedicine and applications*, vol. 2017, 2017.

[10] B. Williams, B. Allen, H. True, N. Fell, D. Levine, and M. Sartipi, "A real-time, mobile timed up and go system," in *Wearable and Implantable Body Sensor Networks (BSN), 2015 IEEE 12th International Conference on*. IEEE, 2015, pp. 1–6.

[11] Q. Li, J. A. Stankovic, M. A. Hanson, A. T. Barth, J. Lach, and G. Zhou, "Accurate, fast fall detection using gyroscopes and accelerometer-derived posture information," in *Wearable and Implantable Body Sensor Networks, 2009. BSN 2009. Sixth International Workshop on*. IEEE, 2009, pp. 138–143.

[12] G. Rescio, A. Leone, and P. Siciliano, "Supervised expert system for wearable mems accelerometer-based fall detector," *Journal of Sensors*, vol. 2013, 2013.

[13] A. Mannini, S. S. Intille, M. Rosenberger, A. M. Sabatini, and W. Haskell, "Activity recognition using a single accelerometer placed at the wrist or ankle," *Medicine and science in sports and exercise*, vol. 45, no. 11, p. 2193, 2013.

[14] A. T. Özdemir and B. Barshan, "Detecting falls with wearable sensors using machine learning techniques," *Sensors*, vol. 14, no. 6, pp. 10 691–10 708, 2014.

[15] J. K. Lee, S. N. Robinovitch, and E. J. Park, "Inertial sensing-based pre-impact detection of falls involving near-fall scenarios," *IEEE transactions on neural systems and rehabilitation engineering*, vol. 23, no. 2, pp. 258–266, 2015.

[16] F. Wu, H. Zhao, Y. Zhao, and H. Zhong, "Development of a wearable-sensor-based fall detection system," *International journal of telemedicine and applications*, vol. 2015, p. 2, 2015.

[17] C. Smeesters, W. C. Hayes, and T. A. McMahon, "Disturbance type and gait speed affect fall direction and impact location," *Journal of biomechanics*, vol. 34, no. 3, pp. 309–317, 2001.

[18] M. Tolkiehn, L. Atallah, B. Lo, and G.-Z. Yang, "Direction sensitive fall detection using a triaxial accelerometer and a barometric pressure sensor," in *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*. IEEE, 2011, pp. 369–372.

[19] I. Cleland, B. Kikhia, C. Nugent, A. Boytsov, J. Hallberg, K. Synnes, S. McClean, and D. Finlay, "Optimal placement of accelerometers for the detection of everyday activities," *Sensors*, vol. 13, no. 7, pp. 9183–9200, 2013.

[20] O. Aziz, E. J. Park, G. Mori, and S. N. Robinovitch, "Distinguishing the causes of falls in humans using an array of wearable tri-axial accelerometers," *Gait & posture*, vol. 39, no. 1, pp. 506–512, 2014.

[21] M. Elhoushi, J. Georgy, A. Noureldin, and M. J. Korenberg, "A survey on approaches of motion mode recognition using sensors," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 7, pp. 1662–1686, 2017.

[22] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[23] I. Kononenko, E. Šimec, and M. Robnik-Šikonja, "Overcoming the myopia of inductive learning algorithms with relieff," *Applied Intelligence*, vol. 7, no. 1, pp. 39–55, 1997.

[24] M. Robnik-Šikonja and I. Kononenko, "Theoretical and empirical analysis of relieff and rrelieff," *Machine learning*, vol. 53, no. 1-2, pp. 23–69, 2003.

[25] F. Nie, H. Huang, X. Cai, and C. H. Ding, "Efficient and robust feature selection via joint â2, 1-norms minimization," in *Advances in neural information processing systems*, 2010, pp. 1813–1821.

[26] A. Liaw, M. Wiener *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[27] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[28] T. S. Furey, N. Cristianini, N. Duffy, D. W. Bednarski, M. Schummer, and D. Haussler, "Support vector machine classification and validation of cancer tissue samples using microarray expression data," *Bioinformatics*, vol. 16, no. 10, pp. 906–914, 2000.

[29] S. R. Gunn *et al.*, "Support vector machines for classification and regression," *ISIS technical report*, vol. 14, pp. 85–86, 1998.

[30] S. Dreiseitl and L. Ohno-Machado, "Logistic regression and artificial neural network classification models: a methodology review," *Journal of biomedical informatics*, vol. 35, no. 5, pp. 352–359, 2002.

[31] M.-L. Zhang and Z.-H. Zhou, "A k-nearest neighbor based algorithm for multi-label classification," in *Granular Computing, 2005 IEEE International Conference on*, vol. 2. IEEE, 2005, pp. 718–721.

[32] M. Mubashir, L. Shao, and L. Seed, "A survey on fall detection: Principles and approaches," *Neurocomputing*, vol. 100, pp. 144–152, 2013.

[33] F. Zubaydi, A. Saleh, F. Aloul, and A. Sagahyroon, "Security of mobile health (mhealth) systems," in *Bioinformatics and Bioengineering (BIBE), 2015 IEEE 15th International Conference on*. IEEE, 2015, pp. 1–5.

[34] J. L. Fernández-Alemán, I. C. Señor, P. Á. O. Lozoya, and A. Toval, "Security and privacy in electronic health records: A systematic literature review," *Journal of biomedical informatics*, vol. 46, no. 3, pp. 541–562, 2013.

VITA

Austin Harris was born in Akron, Ohio, to the parents of Steve and Marlyce Harris. He is the youngest of three children. After graduating from South Side High School in Jackson, TN, he attended the University of Tennessee, Chattanooga. In 2015, he was awarded the Bachelors of Science in Computer Science. He returned to graduate school immediately after receiving his Bachelors degree. Austin graduated with a Masters of Science from the Univeristy of Tennessee, Chattanooga, in Computer Science in December 2017.